Information Theory and Predictability. Lecture 4: Optimal Codes

1. Introduction

We discussed codes briefly in Lecture 1. We now turn to the subject in detail.

Definition 1. An *encoding* $c(x)$ for a random variable X is a mapping from the set of outcomes of the random variable $\{x\}$ to a string of symbols from a finite alphabet of D distinct symbols.

For most of this lecture we will consider for clarity a two member bit alphabet $\{0, 1\}$.

Definition 2. The *extension* of an encoding is the mapping from an ordered string of outcomes to an ordered string of symbols from the code alphabet given by $c(x_1x_2 \dots x_n) \equiv c(x_1)c(x_2) \dots c(x_n)$

Associated with each encoding is a length which is the number of symbols in the code string. We write this as $l(c(x))$. For a particular random variable we can await a series of outcomes and for every one transmit an encoding. This will result in a continually expanding string of symbols given by the code extension defined above. We term this a transmission. Asymptotically i.e. given a very large number of outcomes the average length of this transmission per outcome will approach a limit governed by the probability function of X.

Definition 3. The *expected length* L of an encoding of X with associated probability function $p(x)$ is given by

$$L = \sum_x p(x)l(c(x))$$

In this lecture we are going to compare L to $H_D(X)$. We consider some illustrative examples:

Suppose we have $D = 2$ and the following probabilities and encodings

Outcome	Probability	Encoding
1	$\frac{1}{2}$	0
2	$\frac{1}{4}$	10
3	$\frac{1}{8}$	110
4	$\frac{1}{8}$	111

A quick calculation here shows that $H_2(X) = L = 1.75$. Note that the less probable outcomes are assigned longer codes which seems like a good strategy intuitively for reducing L.

Suppose now we take $D = 3$ and the following setup

Outcome	Probability	Encoding
1	0.2	0
2	0.2	10
3	0.2	21
4	0.2	20
5	0.2	11

It is easy to calculate $H_3(X) \approx 1.465 < L = 1.8$.

Finally consider the case of the Morse code which has $D = 4$ as it has a dash; a dot and two types of spaces (words and letters). Using the set of all books to define the (approximate!) probabilities of letters and spaces in an arbitrary Morse transmission one can in principle calculate $H_4(X)$ as well as L and one finds that the latter is greater. Note that the designers of this code attempted for practical reasons to reduce L by making sure that less likely letters generally have longer sequences of dots and dashes.

Codes are not all equally useful which leads to the following definitions.

Definition 4. 1. An encoding is *non-singular* if the mapping $c(x)$ is one to one i.e. $x \neq x' \Rightarrow c(x) \neq c(x')$. 2. An encoding is *uniquely decodeable* if its extension to any transmission is non-singular. 3. An encoding is *instantaneously decodeable* if the string of characters transmitted is (uniquely) decodeable "on the fly" whenever an additional codeword is transmitted. In other words the code is cumulatively and immediately translated as any new codeword is read.

It is easily checked by example that these three classes of encoding include (properly) the later ones. In addition it is also easily seen that an encoding is instantaneously decodeable iff no code $c(x)$ is a *prefix* of another $c(x')$. This follows because if a codeword is received that is a prefix of another then immediate translation is not possible and one needs to wait to see whether the longer codeword was meant rather than the prefix. Clearly if any codeword is not a prefix of another one need not wait and so the code is instantaneously decodeable. Looking at the first two code examples above we see that they are both instantaneously decodeable since no code is a prefix of another.

2. Basic Coding Results

Clearly from a practical viewpoint a non-singular encoding is essential while an instantaneously decodeable one is desirable. In addition there is a clear utilitarian value in finding an instantaneously decodeable encoding of minimal expected length L.

Regarding such an objective we can establish the following known as the Kraft Inequality.

Theorem 5. *Consider an instantaneously decodeable encoding with a D category alphabet and with code lengths $l_i = l(c(x_i))$ then*

$$\sum_i D^{-l_i} \leq 1 \tag{2.1}$$

Conversely if one finds a set of integers $\{l_i\}$ satisfying (2.1) then there is an instantaneously decodeable encoding with code lengths given by this set.

Proof. We establish the result for the case that the random variable has finite outcomes. The extension to the countably infinite case is fairly straightforward and can be found on page 109 of [1].

Consider a tree with n levels and where each branch point results in D branches. We can use such a tree to enumerate all the codewords if we label each branch set with all D members of our alphabet. Obviously if no codeword is a prefix of another (the condition for the encoding to be instantaneously decodeable) then any node which is a codeword cannot have any codewords at higher levels. Now suppose l_{max}

is the maximum codeword length then the tree at this level must have a total of $D^{l_{max}}$ branches. Consider the codewords at level l_i in the tree. They must have a total of $D^{l_{max}-l_i}$ descendant branches at the l_{max} level of the tree (all of which are of course not codewords because of the assumed prefix property). Also obviously because of the structure of a tree all these descendant branch sets at the l_{max} level must be disjoint. Thus adding up all these disjoint descendants and comparing with the total number of possible branches at the l_{max} level then it is clear that

$$\sum_i D^{l_{max}-l_i} \leq D^{l_{max}} \tag{2.2}$$

which is equivalent to the Kraft inequality. Conversely suppose we have a set of (reverse ordered) $\{l_i\}$ which satisfy equation (2.2) then commencing at the bottom of the set of branches at level l_{max} and proceeding across this tree level we can gather successively $D^{l_{max}-l_1}, D^{l_{max}-l_2}, \ldots, D^{l_{max}-l_N}$ set of branches where N is the total number of codewords. Because of the inequality we never run out of branches. Some thought shows that each of these set of branches has a unique ancestor at the l_i level of the tree which we identify as a codeword by labelling the whole tree in the obvious way. This constructs an encoding for which clearly no code is a prefix of another. □

The Kraft inequality now allows us to establish a lower bound on the expected length L for instantaneously decodeable encodings and also determine when this is reached. The proof is instructive as to the power of the relative entropy functional.

Theorem 6. *Suppose we have an instantaneously decodeable encoding for a random variable X with a B member coding alphabet then the expected length satisfies*

$$L \geq H_B(X)$$

with equality achieved if and only if the probabilities and codeword lengths satisfy

$$p_i = B^{-l_i} \tag{2.3}$$

Proof. The difference Δ between L and $H_B(X)$ is

$$\Delta = \sum_i p_i l_i + \sum_i p_i \log_B p_i = -\sum_i p_i \log_B B^{-l_i} + \sum_i p_i \log_B p_i$$

Defining a new probability $r_i \equiv B^{-l_i} / \sum_j B^{-l_j}$ then we can write

$$\Delta = \sum_i p_i \log_B \frac{p_i}{r_i} - \log_B \sum_j B^{-l_j} = D(p||r) - \log_B \sum_j B^{-l_j}$$

The two terms at the right here are non-negative because of the property of relative entropy as well as the Kraft inequality. They will vanish if and only if $p_i = r_i$ and $\sum_j B^{-l_j} = 1$ again by the properties of relative entropy. □

It is clear that the lower bound will not always be achievable for a given random variable X and coding alphabet B since equation (2.3) may not be satisfiable for any set of l_i. Notice that our first example above does in fact satisfy (2.3) which explains why its expected length is exactly the entropy. There remain therefore two important questions: What exactly is the practical lower bound and how can it be achieved? We begin with the first question.

Theorem 7. *Suppose we have the conditions of the previous theorem then the minimal possible expected length L^* satisfies the inequality*

$$H_B(X) \leq L^* \leq H_B(X) + 1 \tag{2.4}$$

Proof. For each p_i it is obviously possible to find a unique $l_i > 0$ such that $B^{-l_i+1} \geq p_i \geq B^{-l_i}$. The right inequality also implies that the l_i satisfy the Kraft inequality and so correspond with an instantaneously decodeable encoding. Taking logarithms to the base B of both inequalities leads to $\log_B(1/p_i) \leq l_i \leq \log_B(1/p_i) + 1$ and summing this after multiplication by p_i shows that the expected length of this particular encoding satisfies the inequality of the theorem. The optimally short code must as well because of this and the previous theorem. □

Note that the construction in the proof of the last theorem allows one through the tree introduced earlier to construct a code which although not necessarily optimal does satisfy equation (2.4). It is known as the Shannon code.

Thinking back to the Morse Code example introduced above it is rather evident that at times one may not actually have the true probability to hand and may be forced to guess it usually on the basis of earlier observations of the random variable (e.g. using the set of all books in our example). If an incorrect probability for a random variable is assumed then using this to perform encoding will intuitively incur a penalty in the expected length. Roughly speaking this penalty is the relative entropy of the correct and incorrect probability functions. More precisely we have:

Theorem 8. *Suppose we assume the probability is q_i when it is in fact p_i then the resulting Shannon code expected length L will satisfy*

$$H(p) + D(p||q) \leq L \leq H(p) + D(p||q) + 1$$

Proof. It is clear that the Shannon code assigned according to the incorrect q_i has codeword lengths satisfying

$$l_i \geq -\log q_i$$

and so it has expected length

$$L = \sum_i p_i l_i \geq -\sum_i p_i \log q_i = H(p) + D(p||q)$$

and the other reverse inequality can be obtained by using the other inequality from the Shannon code proof. □

The relative entropy has therefore the concrete interpretation as the penalty for incorrectly assuming a particular probability holds when in fact another does. We shall revisit this interpretation later in the course.

As we saw earlier instantaneously decodeable encodings are a subset of uniquely decodeable encodings so one may try to obtain a shorter expected length by relaxing the instantaneous assumption. Rather surprisingly a result due to McMillan [3] shows in fact that such codes also satisfy the Kraft inequality i.e. Theorem 5 and consequently that they offer no further optimality for expected length. This follows because there must exist another instantaneously decodeable code which has the same expected length by the constructive proof of Kraft's inequality. Interested readers can find the rather technical proof in [1].

3. The Huffman Optimal Code

The Shannon code discussed in the previous section is not always optimal in terms of minimizing the expected length. There exists however an algorithm due to Huffman [2] which can be used to construct one particular optimal code (there may be many). In the following we restrict ourselves for clarity to codes with two symbols (bit codes) but the method extends with some minor modification to finite symbol codes. The algorithm works as follows:

Order the outcomes $x_1, x_2, \ldots x_m$ so that $p_1 \leq p_2 \leq \ldots \leq p_{m-1} \leq p_m$ and construct a tree bottom up as follows: Combine the least likely two outcome categories x_1 and x_2 and then reorder the new $m-1$ probabilities. This operation results in a higher tree level with one less category. Repeat the operation to further reduce the number of categories (each reduction is a higher tree level) and iterate until only one category remains containing all outcomes. Looking back down the complete tree for every split of categories assign an additional 1 to all outcomes from one of the newly split category and a 0 to all the other outcomes from the other newly split category. Continue down to the bottom of the tree where all splits are done and one has recovered all the original categories. A Huffman codeword is profitably viewed as a set of instructions as to which branch to take moving down from the 1 node "tree top" with instructions starting at the left. Thus a 1 says take the left branch (when a split occurs) while a 0 says take the right branch. The instructions finish when the branch taken hits the m outcome level. Obviously these instructions (and hence the codeword) point to a unique outcome. Additionally suppose one codeword is a prefix of another then the first set of instructions imply the branch trip finishes at level m but the second codeword implies the this same trip should have ended at level j where $j < m$ which is obviously impossible so the Huffman code is prefix free.

We now show that the Huffman encoding is optimal.

Lemma 9. *There exists an optimal instantaneous encoding termed a canonical optimal encoding with the following three properties*

(1) If $p_j > p_k$ then $l_j \leq l_k \quad \forall j, k$
(2) The longest codewords are equal in length.
(3) Two of the longest codewords differ only in the final bit and correspond with the two least likely outcomes.

Proof. Exchange codewords for j and k. It is easy to check that the new code will have an additional expected length of $(p_j - p_k)(l_k - l_j)$ which shows that if property 1 did not hold then the new code would have a shorter length implying that the original code could not be optimal.

Secondly if the longest two codewords were not equal in length we could trim the longest so they were and not violate the prefix property as well as producing a shorter code. This is therefore not possible if the code is optimal.

Thirdly observe that any of the longest codewords of length n must have a "sibling" in the sense that the two share an $n-1$ character prefix. If that was not the case we could truncate such a codeword by deleting the last bit and still retain the prefix property and hence produce a new instantaneous code with shorter expected length. We can now shuffle all the codewords of length n between outcomes without changing the expected length of the code. We can do this in order to ensure that the least likely outcomes are paired (i.e. share an $n-1$ character prefix) which ensures

that the last part of property 3 holds. The least likely outcomes must be have the longest length n otherwise property 1 won't hold. Notice that this shuffling without effect on expected length results in many different optimal codes not all canonical because property 3 does not always hold. □

We can now prove the main result:

Theorem 10. *The Huffman encoding C^* is optimal in the sense that if C' is any other uniquely decodeable encoding then $L(C^*) \leq L(C')$ where the functional L is the expected length of an encoding.*

Proof. By induction. First observe that earlier results allow us to restrict our attention to instantaneously decodeable codes.

Second suppose we have the following n member ordered probabilities $\mathbf{p} \equiv \{p_1, p_2, \ldots, p_n\}$ and the associated $n-1$ member $\mathbf{p}' \equiv \{p_1, p_2, \ldots, p_{n-1} + p_n\}$. Suppose we have an optimal encoding defined on $\mathbf{p}'$ and call it $C_1(\mathbf{p}')$ and further a canonical optimal code on $\mathbf{p}$ denoted by $C_2(\mathbf{p})$. We can extend the C_1 from $\mathbf{p}'$ to $\mathbf{p}$ by adding two different bits to the code for $p_{m-1} + p_m$. Call this new code $C_3(\mathbf{p})$. We can also restrict the code C_2 from $\mathbf{p}$ to $\mathbf{p}'$ by taking the codewords for p_{n-1} and p_n and removing their last bit. This results in a unique new code $C_4(\mathbf{p}')$ by the third property of the lemma above. Both new codes are still instantaneously decodeable by their construction. Simple calculation shows that $L(C_3) = L(C_1) + p_{n-1} + p_n$ and $L(C_4) = L(C_2) - p_{n-1} - p_n$ which shows that

$$L(C_4) - L(C_1) = L(C_2) - L(C_3)$$

since C_1 and C_2 are optimal by assumption then both sides of this equation must be zero implying that the extension code C_3 must also be optimal for $\mathbf{p}$. Thus we have a well defined method for preserving optimality as we split outcome classes as we do in the Huffman algorithm. Since an optimal code for two outcomes is obvious (and the first step of the Huffman algorithm) we are done by induction. □

References

[1] T.M. Cover and J.A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, 2nd edition, 2006.

[2] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40:1098–1101, 1952.

[3] B. McMillan. Two inequalities implied by unique decipherability. *IEEE Trans. Inf. Theory*, IT-2:115–116, 1956.